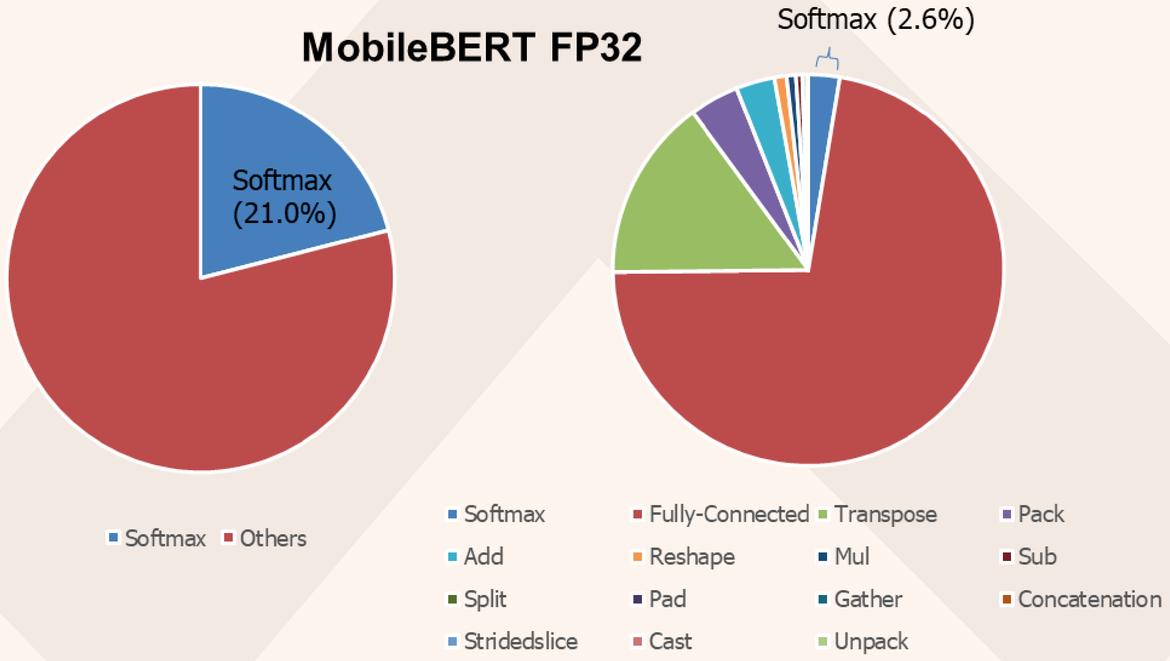
MobileBERT On RISC-V: Leveraging IREE Compiler and ACE-RVV Extension for Softmax Acceleration

Yueh-Feng Lee, Heng-Kuan Lee, Yi-Jui Chu
Andes Technology

Visit Andes' website for more info

Motivation

- IREE softmax RVV codegen is not as efficient as manually optimized library
- Nonlinear functions based on the exponential function



- Softmax

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

- Tanh

$$\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- GeLU

$$GeLU(x) = 0.5 \times x \times \left(1 + \tanh\left(\sqrt{(2/\pi)} \times (x + 0.044715 \times x^3)\right)\right)$$

- SWiGLU

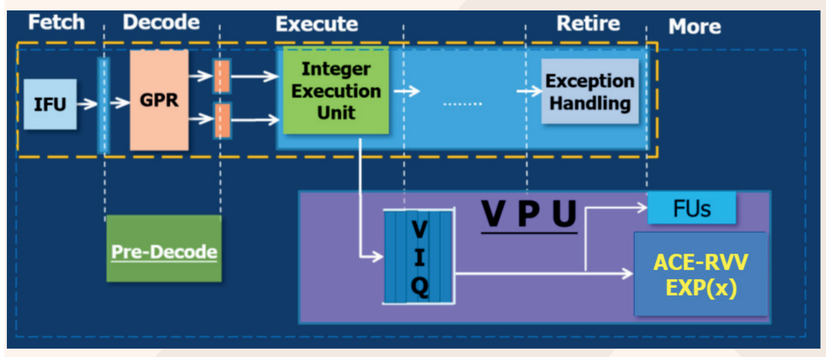
$$Swish(x) = x \cdot sigmoid(\beta x) = x/\left(1 + e^{-\beta x}\right)$$

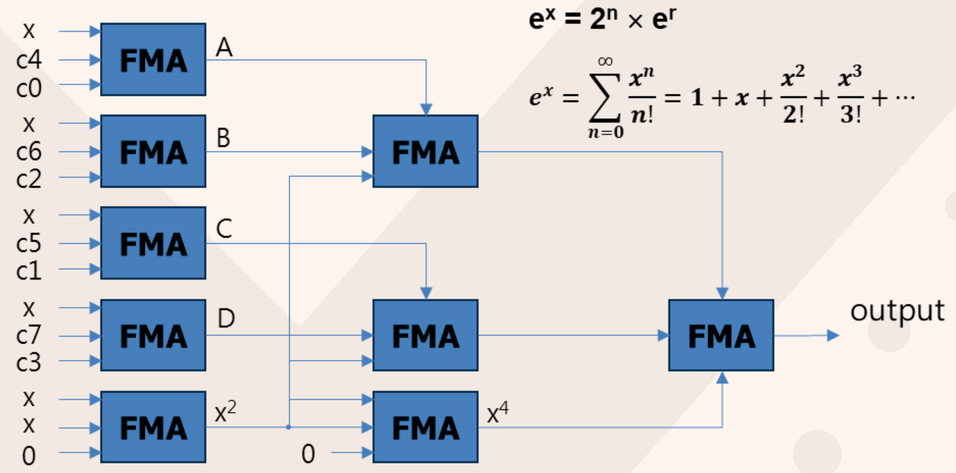
$$SwiGLU(x, W, V, b, c, \beta) = Swish_\beta(xW + b) \otimes (xV + c)$$

ACE-RVV Instruction Design

- Custom Exp Instruction Using ACE-RVV
- Exp Instruction by 7-Order Taylor Expansion

ACE-RVV: Reuse processor memory system





$$e^x = 2^n \times e^r$$

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

➢ Optimized Exp Function

Target	Throughput (Ratio)	Precision (SNR dB)
RVV + Exp	7.31	>117.08
RVV	1.00	>117.08

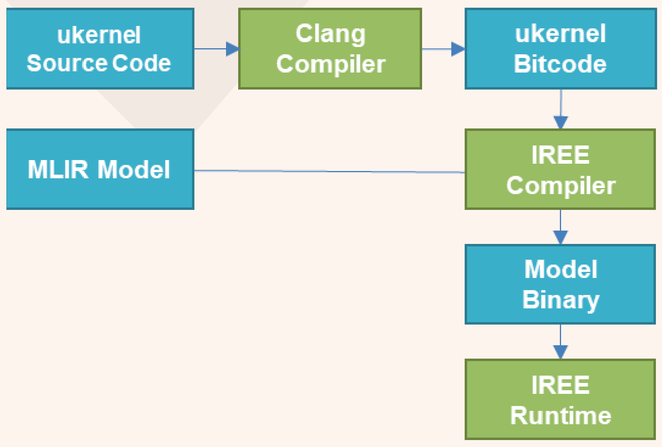
➢ Optimized Softmax Function

Target	Throughput (Ratio)
RVV + Exp	4.66
RVV	1.00

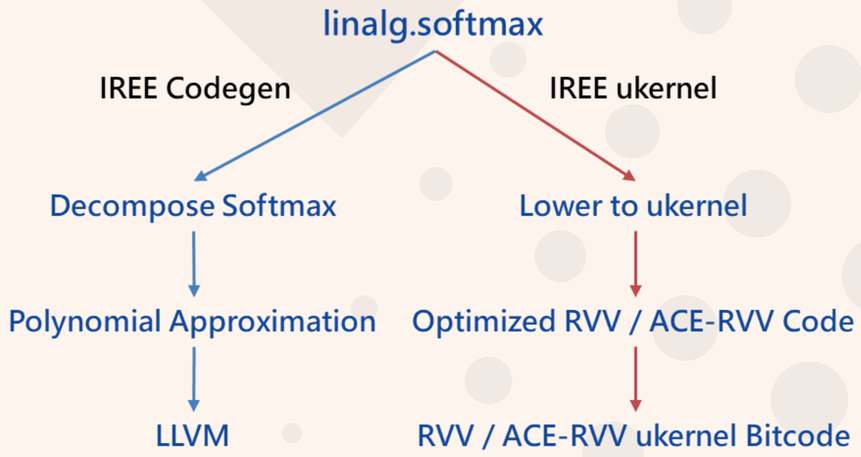
AX45MPV VLEN/DLEN = 512/512
- L1 data cache: 32KB, instruction cache: 32KB
- L2 cache: 512KB

IREE Softmax ukernel Integration

IREE ukernel Flow



IREE with Andes Softmax ukernel

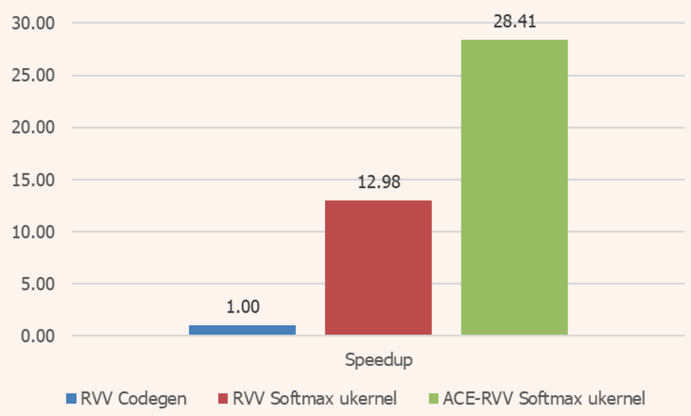


IREE Decompose Softmax

```
%6 = linalg.generic {indexing_maps = [affine_map<(d0, d1, d2, d3) -> (d0, d1, d2, d3)>, affine_map<(d0,
d1, d2, d3) -> (d0, d1, d2)>], iterator_types = ["parallel", "parallel", "parallel", "reduction"]} ins(%2
: tensor<1x4x384x384xf32>) outs(%5 : tensor<1x4x384xf32>) {
  ^bb0(%in: f32, %out: f32):
    %10 = arith.maximumf %in, %out : f32
    linalg.yield %10 : f32
  } -> tensor<1x4x384xf32>
  %cst_0 = arith.constant 0.000000e+00 : f32
  %7 = linalg.fill ins(%cst_0 : f32) outs(%4 : tensor<1x4x384xf32>) -> tensor<1x4x384xf32>
  %8 = linalg.generic {indexing_maps = [affine_map<(d0, d1, d2, d3) -> (d0, d1, d2, d3)>, affine_map<(d0,
d1, d2, d3) -> (d0, d1, d2)>, affine_map<(d0, d1, d2, d3) -> (d0, d1, d2)>], iterator_types = ["parallel",
"parallel", "parallel", "reduction"]} ins(%2, %6 : tensor<1x4x384x384xf32>, tensor<1x4x384xf32>) outs(%7 :
tensor<1x4x384xf32>) {
  ^bb0(%in: f32, %in_1: f32, %out: f32):
    %10 = arith.subf %in, %in_1 : f32
    %11 = math.exp %10 : f32
    %12 = arith.addf %11, %out : f32
    linalg.yield %12 : f32
  } -> tensor<1x4x384xf32>
  %9:2 = linalg.generic {indexing_maps = [affine_map<(d0, d1, d2, d3) -> (d0, d1, d2, d3)>,
affine_map<(d0, d1, d2, d3) -> (d0, d1, d2)>, affine_map<(d0, d1, d2, d3) -> (d0, d1, d2)>,
affine_map<(d0, d1, d2, d3) -> (d0, d1, d2, d3)>, affine_map<(d0, d1, d2, d3) -> (d0, d1, d2, d3)>],
iterator_types = ["parallel", "parallel", "parallel", "parallel"]} ins(%2, %6, %8 :
tensor<1x4x384x384xf32>, tensor<1x4x384xf32>, tensor<1x4x384xf32>) outs(%3, %3 : tensor<1x4x384x384xf32>,
tensor<1x4x384x384xf32>) {
  ^bb0(%in: f32, %in_1: f32, %in_2: f32, %out: f32, %out_3: f32):
    %10 = arith.subf %in, %in_1 : f32
    %11 = math.exp %10 : f32
    %12 = arith.divf %11, %in_2 : f32
    linalg.yield %11, %12 : f32, f32
  } -> (tensor<1x4x384x384xf32>, tensor<1x4x384x384xf32>)
}
```

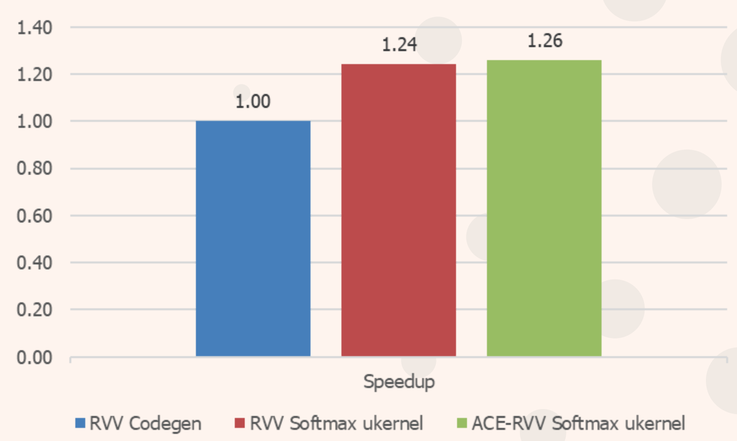
Experimental Results

Softmax Unit Test Speedup

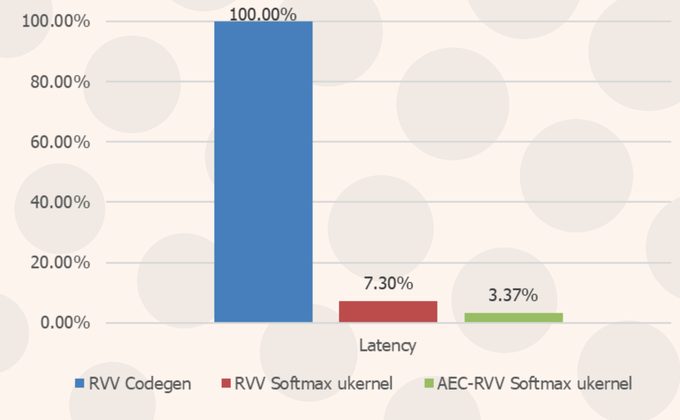


Shape 1x4x384x384xf32

MobileBERT FP32 Speedup



MobileBERT Softmax Op Latency



AX45MPV VLEN 512 8M L2

- **Using IREE with Andes ACE-RVV softmax ukernel, MobileBERT has achieved a 1.26x speedup, and softmax latency has been reduced to 3.37%.**